Computational Semantics

Assignment 5: Tableaux and Model Checking
Assignment 6: Resolution and Theorem Proving

Assignment 4.2.1

$\neg\neg p \to p$

1. $F(\neg\neg p \to p)$		✓
2. $T(\neg\neg p)$	1, $F\to$	✓
3. $F(p)$	1, $F\to$	
4. $F(\neg p)$	2, $T\neg$	✓
5. $T(p)$	4, $F\neg$	

Unable to falsify, holds in all cases and thus a validity.

$((p \to q) \to p) \to p$

1. $F(((p \to q) \to p) \to p)$		✓
2. $T((p \to q) \to p)$	1, $F\to$	✓
3. $F(p)$	1, $F\to$	✓

4. $F(p \to q)$	2, $T\to$ ✓	5. $T(p)$	2, $T\to$
6. $T(p)$	4, $F\to$		
7. $F(q)$	4, $F\to$		

Unable to falsify because both branches require p to be true, while p had to be false even before the branching.

$(\neg p \to \neg q) \to (q \to p)$

1. $F((\neg p \to \neg q) \to (q \to p))$		✓
2. $T(\neg p \to \neg q)$	1, $F\to$	✓
3. $F(q \to p)$	1, $F\to$	✓
4. $T(q)$	3, $F\to$	
5. $F(p)$	3, $F\to$	

6. $F(\neg p)$	2, $T\to$ ✓	7. $T(\neg q)$	2, $T\to$ ✓
8. $T(p)$	6, $F\neg$	9. $F(q)$	7, $T\neg$

Unable to falsify. Before the branching it has been established that q has to be true and p has to be false. In the first branch p has to be true and in the second branch q has to be false, which contradicts what has been said before and thus the model is valid.

$p \to (p \wedge (q \vee p))$

1. $F(p \to (p \wedge (q \vee p)))$		✓
2. $T(p)$	1, $F\to$	
3. $F(p \wedge (q \vee p))$	1, $F\to$	✓

4. $F(p)$	3, $F\wedge$	5. $F(q \vee p)$	3, $F\wedge$ ✓
		6. $F(q)$	5, $F\vee$
		7. $F(p)$	5, $F\vee$

Unable to falsify because before the branching p had to be true and in both branches p has to be false.

(p∨(q∧r))→((p∨q)∧(p∨r))

1. F((p∨(q∧r))→((p∨q)∧(p∨r)))	✓
2. T(p∨(q∧r))	1, F→ ✓
3. F((p∨q)∧(p∨r))	1, F→ ✓

4. T(p)	2, T∨			5. T(q∧r)	2, T∨ ✓		
				6. T(q)	5, T∧		
				7. T(r)	5, T∧		
8. F(p∨q)	3, F∧ ✓	9. F(p∨r)	3, F∧ ✓	10. F(p∨q)	3, F∧ ✓	11. F(p∨r)	3, F∧ ✓
12. F(p)	8, F∨	13. F(p)	9, F∨	14. F(p)	10, F∨	15. F(p)	11, F∨
16. F(q)	8, F∨	17. F(r)	9, F∨	18. F(q)	14, F∨	19. F(r)	11, F∨

Unable to falsify because:
In branch one, step 4 dictates p has to be true while in step 12 p has to be false
In branch two, step 4 dictates p has to be true while in step 13 p has to be false
In branch three, step 6 dictates q has to be true while in step 18 q has to be false
In branch four, step 7 dictates r has to be true while in step 19 r has to be false

Assignment 4.4.1

¬¬p→p
Negate:
1. ¬(¬¬p→p)
Convert tot NNF:
2. ¬(p→p) (Remove double negations)
3. p∧¬p (Eliminate implications)
Convert to CNF:
4. [[p],[¬p]]

((p→q)→p)→p
Negate:
1. ¬(((p→q)→p)→p)
Convert to NNF:
2. ((p→q)→p)∧¬p (Eliminate implications)
3. (¬(p→q)∨p)∧¬p (Eliminate implications)
4. (p∧¬q∨p)∧¬p (Eliminate implications)
Convert to CNF:
5. ((p∨p)∧(¬q∨p))∧¬p (Distribution rule)
6. [[p,p],[¬q,p],[¬p]]

(¬p→¬q)→(q→p)
Negate:
1. ¬((¬p→¬q)→(q→p))
Convert to NNF:
2. (¬p→¬q)∧¬(q→p) (Eliminate implications)
3. (¬p→¬q)∧(q∧¬p) (Eliminate implications)
4. (¬¬p∨¬q)∧(q∧¬p) (Eliminate implications)
5. (p∨¬q)∧(q∧¬p) (Remove double negations)
Convert to CNF:
6. [[p,¬q],[q],[¬p]]

p→(p∧(q∨p))
Negate:
1. ¬(p→(p∧(q∨p)))
Convert to NNF:
2. p∧(¬(p∧(q∨p))) (Eliminate implications)
3. p∧(¬p∨¬(q∨p)) (Drive negations inwards)
4. p∧(¬p∨(¬q∧¬p)) (Drive negations inwards)
Convert to CNF:
5. p∧((¬p∨¬q)∧(¬p∨¬p)) (Distribution rule)
6. [[p],[¬p,¬q],[¬p,¬p]]

(p∨(q∧r))→((p∨q)∧(p∨r))
Negate:
1. ¬((p∨(q∧r))→((p∨q)∧(p∨r)))
Convert to NNF:
2. (p∨(q∧r))∧(¬((p∨q)∧(p∨r))) (Eliminate implications)
3. (p∨(q∧r))∧(¬(p∨q)∨¬(p∨r)) (Drive negations inwards)
4. (p∨(q∧r))∧((¬p∧¬q)∨(¬p∧¬r)) (Drive negations inwards)
Convert to CNF:
5. ((p∨q)∧(p∨r))∧((¬p∧¬q)∨(¬p∧¬r)) (Distribution rule)
6. ((p∨q)∧(p∨r))∧((¬p∨(¬p∧¬r))∧(¬q∨(¬p∧¬r))) (Associativity rule)
7. ((p∨q)∧(p∨r))∧((¬p∨¬p)∧(¬p∨¬r)∧(¬q∨¬p)∧(¬q∨¬r)) (Distribution rule)
8. [[p,q],[p,r],[¬p,¬p],[¬p,¬r],[¬q,¬p],[¬q,¬r]]

Exercise 4.4.5

Starting point is the CNF from previous exercise, if needed a conversion to set CNF is done and finally the resolution rule is used for all clause sets.

[[p],[¬p]]
[[]] —> one empty clause

[[p,p],[¬q,p],[¬p]]
Conversion to set CNF:
[[p],[¬q,p],[¬p]]
[[],[¬q,p]] —> one empty clause

[[p,¬q],[q],[¬p]]
[[],[]] —> two empty clauses

[[p],[¬p,¬q],[¬p,¬p]]
Conversion to set CNF:
[[p],[¬p,¬q],[¬p]]
[[],[¬p,¬q]] —> one empty clause

[[p,q],[p,r],[¬p,¬p],[¬p,¬r],[¬q,¬p],[¬q,¬r]]
Conversion to set CNF:
[[p,q],[p,r],[¬p],[¬p,¬r],[¬q,¬p],[¬q,¬r]]
[[],[],[¬p],[¬q,¬r]] —> two empty clauses

Axioms in first-order logic for spatial relations
The spatial relations used in the image models are:
- Near
- Part_of
- Supports
- Touch

Near is symmetric
Touch is symmetric
Near and touch are disjunct
Supports is conjunct with touch
Thus, near and supports are disjunct
Part_of is disjunct with supports and touch
If something is part of another thing, the near relation holds for both